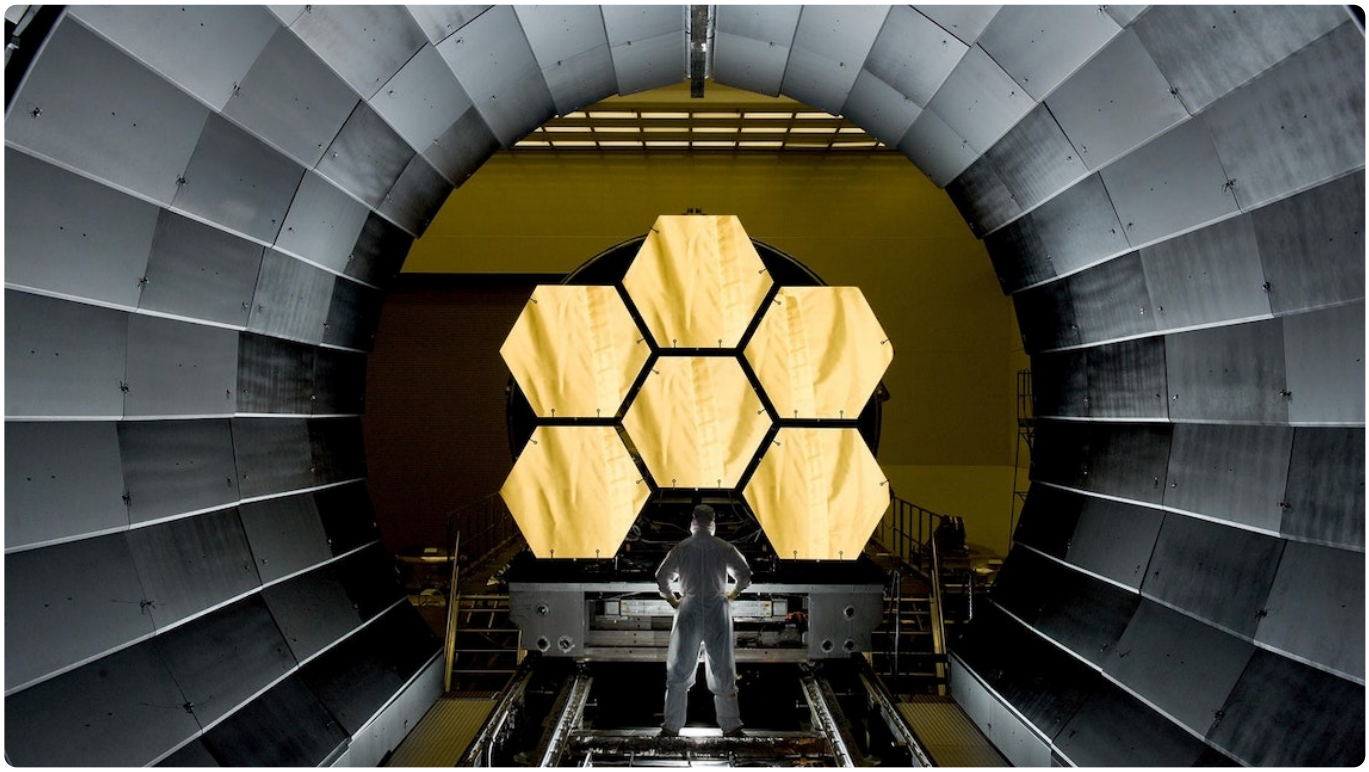
25 | 如何在生产环境中进行真实的容量测试?

2020-01-22 庄振运

性能工程高手课 进入课程 >



讲述：庄振运
时长 19:35　大小 15.70M

你好，我是庄振运。

从今天开始，我们进入新的模块：性能工程实践。在这一模块中，我会讲述在实际生产环境中应用性能工程的场景、案例。这些场景和案例都是针对大规模互联网服务，是在解决实际性能问题后总结的经验。

今天我要讲的主题是“在生产环境中进行真实场景的压力测试”。这来源于我对 LinkedIn 公司生产实践的总结。

LinkedIn 为超过 5.9 亿用户提供服务，在性能优化的过程中，经常会遇到这类问题：一个服务可以承受的最大 QPS 是多少？要满足 100K QPS 的服务需求，我需要多少服务

器？……

怎么解决这些问题呢，有一个大招就是在生产环境中进行真实的容量测试。

关于这个实践的详细方案和技术细节，我们曾经发表过一篇研究论文（IEEE ICWS），并且很荣幸地获得了 IEEE 最佳论文奖，推荐你去读一读。

今天我就带你从这个场景的源头出发，一步步探索到最后的具体落地方案。

为什么需要在生产环境中进行容量测试？

既然我们要解决一个具体的问题，那我们一定要先问个“为什么”。

为什么要在生产环境中进行容量测试呢？要回答这个问题，我们还得再往前追问，为什么说真实的容量测试很重要呢？

我想这个问题不难回答，你应该已经有自己的答案了。

我们都知道，一个在线互联网公司的存活和发展，靠的是它提供的互联网在线服务，自然也就依赖于这些服务的性能和稳定性。要保证每个服务都能够稳定地运行，我们必须为之提供足够的服务容量，比如适当数量的服务器。

那么要如何保证服务容量足够呢？首先就必须做好**服务容量的预测**。

要预测服务容量，我们就需要做容量的性能测试。一般是先确定每台单独的服务器可以支撑多少服务流量；然后用这个单台服务器的数据，来决定这个服务整体需要多少台服务器。这种测试其实就是我们前面讲过的容量测试。

举个具体例子，如果测试结果表明，一台服务器最多可以支撑 100 个 QPS，那么要满足 100K QPS 的服务需求，总共就需要部署一千台服务器。

讲到这里，我想再补充几句。我们讲了这么多服务容量的事儿，那“服务”到底都指的是什么呢？

一般来说，公司提供的服务大致上分为两种：前端服务和后端服务。前端服务是什么样呢？包括各种服务的登陆页面和移动 App，这些服务会直接影响用户的体验。后端服务呢，一般是为前端服务和其他后端服务提供数据和结果。后端服务可以有多种，比如键值数据存储服务（例如，Apache Cassandra），和公司内部的各种微服务。

服务容量预测的重要性我们了解了，那测试为什么需要在生产环境中进行呢？

你是不是想到了，在非生产环境中的容量测试，执行起来肯定更简单啊！没错，的确会更简单，但是，在实验环境或者其他非生产环境中做这样的测试，比如采用人工合成的流量负载，会非常不准确。

这是因为实际的生产环境里面，有多个特殊因素会导致和非生产环境中不同的结果。比如：

1. 客户需求会随着时间而变化，例如高峰时段与非高峰时段的流量就很不一样；
2. 用户请求的多样性，例如不同国家的查询类型不同；
3. 负载流量的规模，基础架构设施的变化，例如服务的软件版本更新，微服务互相调用的变化等等。

所以，对一个重要而复杂的互联网在线服务，由于难以在非生产环境中进行准确的容量测试，我们经常需要转向真正的生产环境，使用实时而真实的客户流量负载来测试。

所以，想要在非生产环境中进行准确的容量测试基本上是做不到的。而对一个重要而复杂的互联网在线服务，能够做到准确的容量测试又太重要了。因为准确的容量数据，是保证线上服务的可靠运行和控制公司成本的基础。

那怎么办呢？这时候我们就需要转向真正的生产环境，使用实时而真实的客户流量负载来测试。

如何在真实生产环境中进行容量测试？

那在真实生产环境中进行容量测试，要如何做呢？

一般来说，我们需要把生产环境的流量进行重定向，让这些重定向的流量，实时地驱动运行 SUT 的单个或者几个服务器。根据重定向的流量大小，会产生不同级别的流量负载。通过

仔细地操作重定向的多少，并且把握测试的时间，我们可以获得非常准确的运行 SUT 服务器的容量值。

但是，生产环境中的容量测量也有诸多挑战，你需要特别小心。

第一个挑战是需要**设计一个控制重定向流量的机制**。这个机制要能够根据其他一些参数，来调整重定向的流量多少。

第二个挑战（也是最大的挑战）是这种**重定向生产流量可能会影响真正的客户**。因为我们会把客户请求重定向到某个服务器，并且会不断给服务器加压，直到这个服务器接近超载，那么这个服务器上所有的客户请求的延迟都会受影响，也就是可能会变大，用户性能也就可能会受到损害。

为了尽量减少对客户的影响，我们的容量测试需要设计合适的机制，来将这种可能的损害降到最低点。系统里面必须有一个模块，来不断地监测客户的性能；一旦到达临界点，就停止继续加压的操作，甚至适当减压。这就是所谓的“非侵入性”（Non-Intrusive）。

还有一个挑战是**对于测试时间的控制**。既然重定向生产流量可能影响客户性能，当然是测试的时间越短越好。可是测试时间太短的话，又可能会影响数据的稳定性。

为了应对这些挑战，准确地确定服务容量的极限，并精确定位容量瓶颈，LinkedIn 采用了一个解决方案，我们将它命名为 RedLiner。

宏观来讲，Redliner 是固定一个生产环境中的 SUT，这个 SUT 包括服务器和上面运行的被测服务，然后不断地把其他服务器上的流量，重定向到这个 SUT 服务器上面。随着流量的不断增大，这个 SUT 服务器的资源使用也就越来越多，所服务的客户请求的性能，比如端到端延迟，就会越来越差。

直到客户请求的性能差到一个定好的阈值，比如端到端延迟是 200 毫秒，流量重定向才会停止。这个时候，基本就可以确定 SUT 服务器不能再处理任何额外的负载。此时获得的容量结果就是 SUT 服务器的最大容量。

想要实现上述的测试，需要好几个模块一起合作。不过在讲 RedLiner 具体的各个模块之前，我们要先梳理一下合理方案的设计原则，一共是五条：

1. 要使用**实时流量**以确保准确性。
2. 尽量不影响生产流量，这就需要**实时的监测和反馈**模块。
3. 可以**定制重定向**等行为规则；对不同的服务和不同的场景的测试，各种性能指标和阈值都会不同。
4. 能**自动终止测试**，并把**测试环境复位**，尽量减少人工干预。
5. 支持基于日历和事件的**自动触发和调度**。

那这个方案具体是怎么实现的呢？

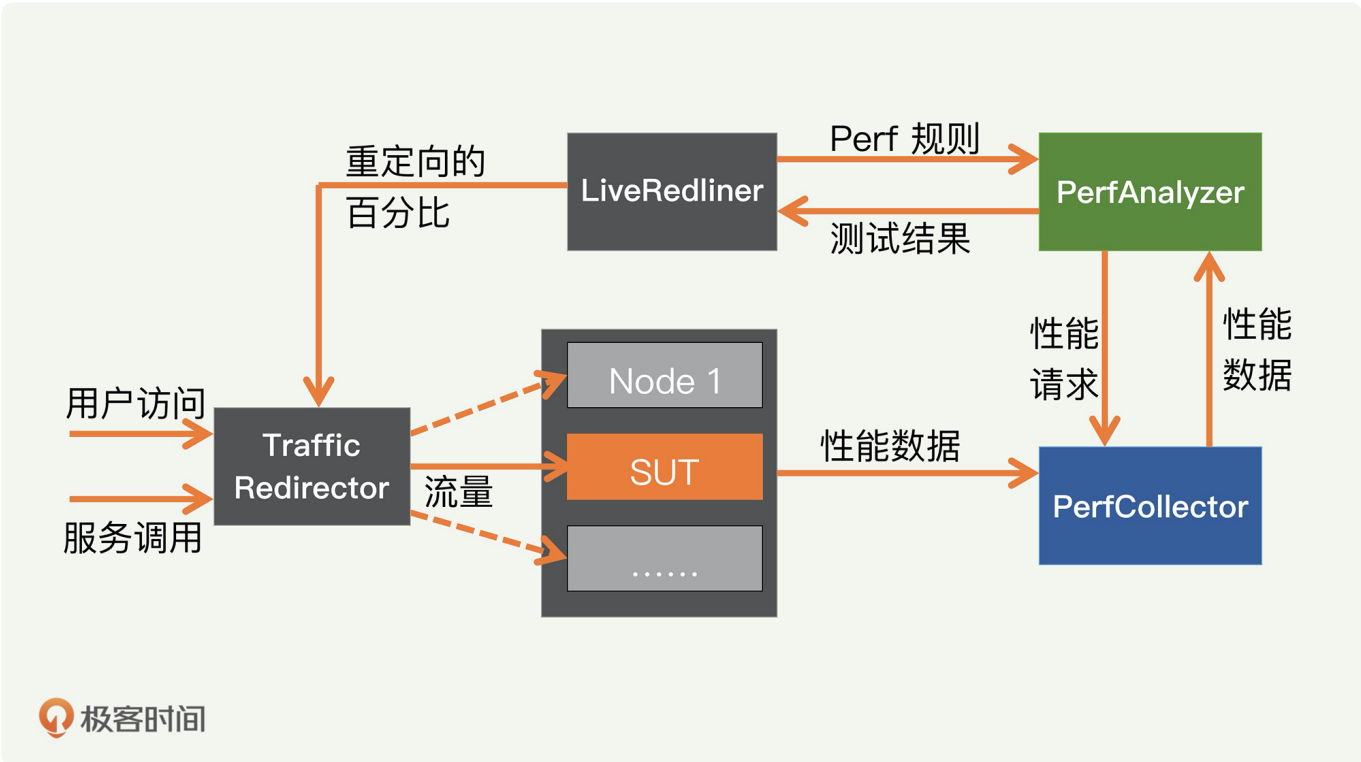
现在我们来看看解决方案的高层架构，如下图所示，这个方案主要包括四个组件：

核心控制器 LiveRedliner

重定向流量的 TrafficRedirector

收集性能数据的 PerfCollector

分析性能数据的 PerfAnalyzer



我先来一一给你介绍下这四个关键组件。

1.**核心控制组件 LiveRedliner**

核心控制组件是整个系统的神经中枢，负责总体调度容量测试的过程。比如何时发起测试，何时终止测试，何时需要增加更大的流量等等。

用户可以自己定义一些特殊的规则，来更好地控制整个测试。用户可以自定义性能指标的阈值。这些性能指标，可以是用户的端到端延迟，包括各种统计指标，比如 P99。也可以把几个不同的性能指标组合起来，实现复杂的逻辑，比如“端到端延迟不超过 200 毫秒，并且错误率不超过 0.1% 等”。

2.**重定向流量组件 TrafficRedirector**

重定向流量组件负责对生产流量进行重定向。具体的实现机制，根据被测试的服务类型分为两种：前端服务和后端服务。

用于**前端服务**时，Redliner 是通过客户请求的属性（例如用户 ID、语言或帐户创建日期），来决定是否对一个客户请求来进行重定向的。这个转换也很简单，可以是取模机制。

举个例子来说，如果 Redliner 需要重定向 1%的流量，它可以把用户的一个属性比如 userid 转换成整数，然后执行用 100 来取模的操作，并且和一个固定整数值作比较。

用于**后端服务**时，重定向流量可以通过另外一个叫做“资源动态发现和负载均衡”的模块来实现。在 LinkedIn，我们很多的服务负载均衡机制，一般会采用一个服务器列表（URI 集群），以相对应的权重值来决定一个请求发送到哪个服务器。

假设这样一个机制有 10 个可用的 URI 集群，并且最初所有这些集群都接收等量的流量（即每个 URI 的权重为 10%）。如果 Redliner 决定将 20%的流量重定向到特定的 URI（即 SUT），那么它可以为 SUT 的 URI 分配 20%的权重。

3.**性能数据收集组件 PerfCollector**

容量测试必须采集各种类型的性能指标，例如 CPU，内存和 QPS 等。这些性能指标的作用，就是确定 SUT 何时达到其容量最大值，以及容量值是多少。

PerfCollector 组件负责收集各种性能指标，包括系统级和服务级的指标。这个组件运行在所有受监视的节点。组件传递的数据量通常很大，因为一般要监测较多的性能指标。所以最好采用扩展性好的实时消息传递系统，来把这些性能数据及时传到其他组件，尤其是下面要介绍的性能分析组件 PerfAnalyer。

我们采用的消息传递系统是 Kafka。Kafka 也是由 LinkedIn 设计并开源的，扩展性和性能都很好，建议你也尝试采用。

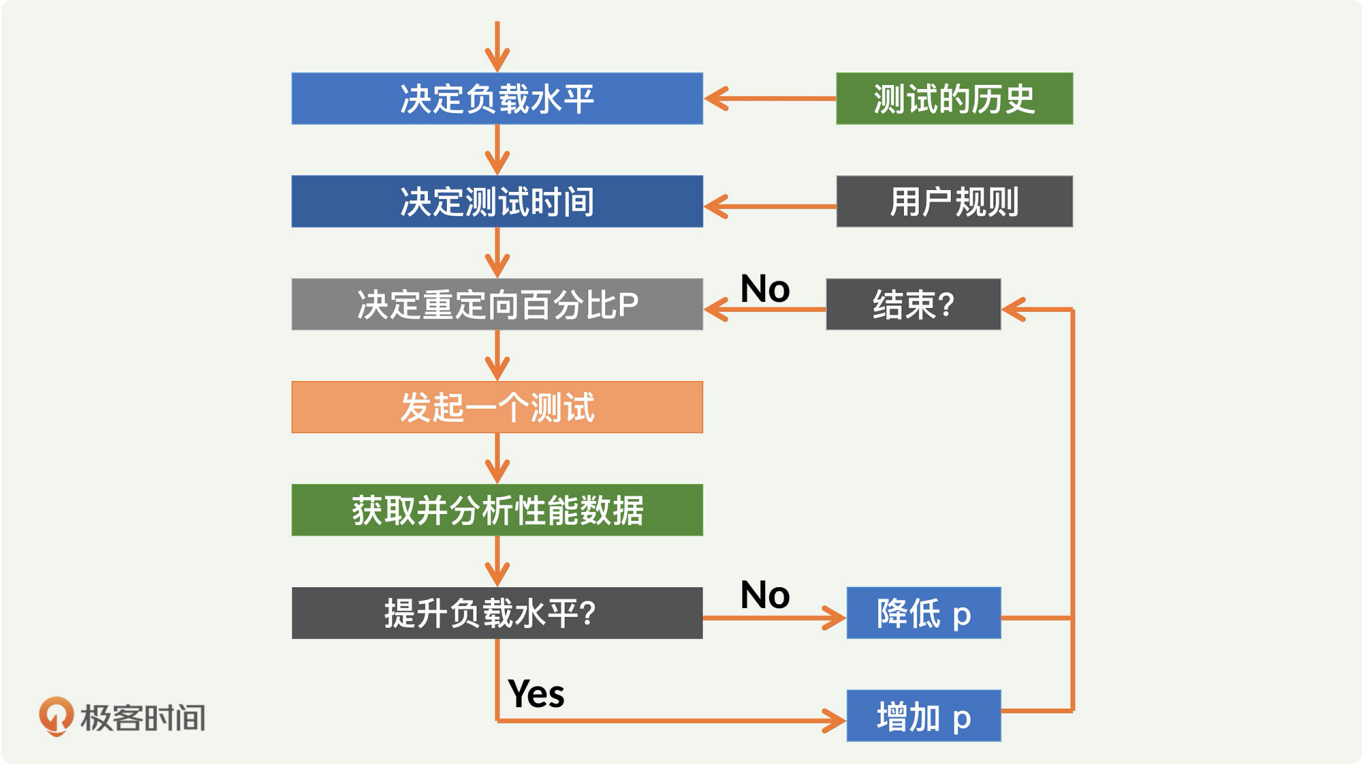
4.**性能分析组件 PerfAnalyzer**

收集性能指标是第一步，下一步就是分析性能，并采取相应的措施。具体来说，可以根据性能指标的值来确定 SUT 是否饱和。

根据性能数据和用户定义的规则，如果发现当前的 SUT，仍有空间来承担更多的负载，我们可以将更大比例的实时流量重定向到该 SUT。否则，如果 SUT 显示饱和迹象，那么重定向的流量百分比就应该降低。PerfAnalyzer 组件能分析收集的数据，并确定是否有特定指标是否违反用户定义的规则。

Redliner 作为一个完整的解决方案，通过几个模块互相配合来实现真实的线上容量测试。通过动态地调整线上的流量，直到让被测系统临近超载状态，从而获得准确的单位服务容量。

现在我们再来换个角度，看一下 RedLiner 的具体操作流程。流程图如下所示。



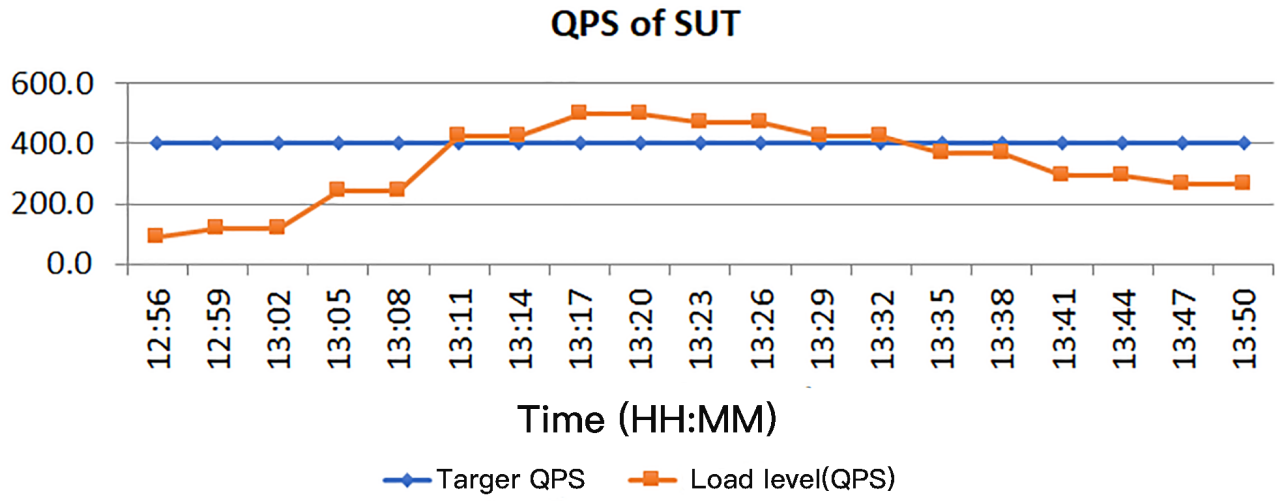
首先是**跟据以前的测试历史数据，决定一个初始重定向数量，也就决定了 SUT 的初始负载**。理论上讲，每个测试都可以从 0 开始；但是如果起点流量太低，整个测试需要花很长时间，所以最好能利用历史数据，从一个比较高的起点开始。

在使用历史数据的基础上，我们还需要**决定总体测试时间**。决定了时间后，每次调整重定向流量的百分比也就确定了。这个调整的数值大小也就是所谓的“步距”，如同人迈步走路，每一步都有大小。对每个百分比，我们一般固定测试 3 分钟，让数据稳定下来，然后调用 PerfAnalyzer 分析并决定下一步。

举个例子，假如我们决定总体测试 60 分钟，并且是从 0% 开始。因为每一个百分点需要测试 3 分钟，那么我们就会决定调整的“步距”大小是 5%，因为最大就是 100%。如果 PerfAnalyzer 决定需要继续增加或者减少重定向百分比，那么就按照前面决定的步距，进行相应的调整。

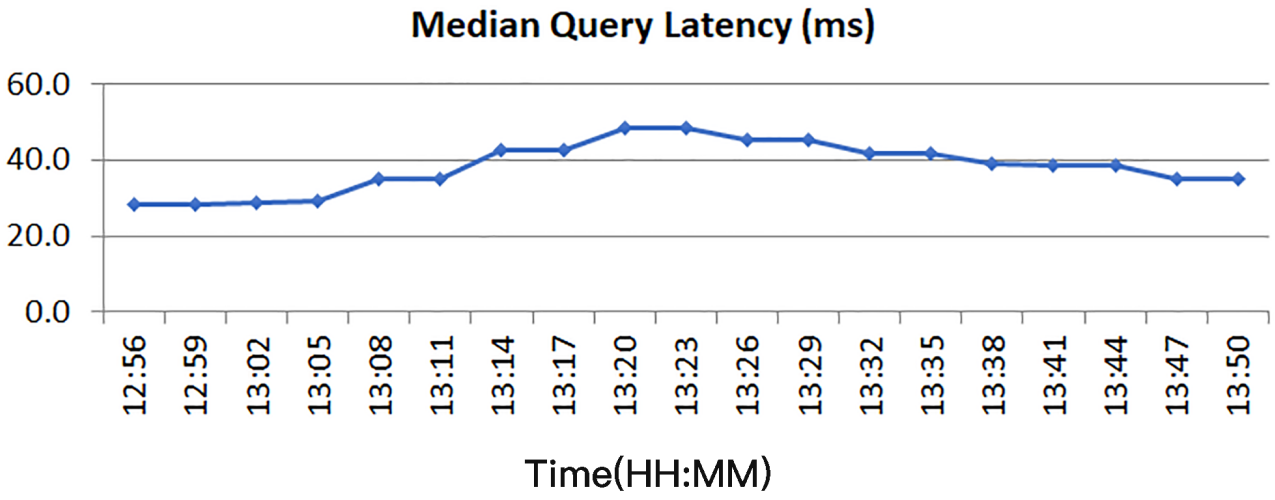
一起来看两个生产环境的数据

刚才讲了一大堆如何实现这个解决方案的内容，现在我们来看看两个生产环境中的实际数据，来直观地感受一下这个系统的特点。下图显示了典型 Redliner 运行的特征。



这是一次完整的容量测试，持续了一个小时。

第一幅图是 QPS，也就是系统吞吐量。蓝色线，表示 SUT 有望实现的目的 QPS。红色线，是 SUT 实现的实际 QPS。我们可以看到，实际的 QPS 值持续变化，这个变化表明了 Redliner 的控制和探测过程，就是一直在动态增加和减少重定向流量百分比。



第二幅图，标识出了所测量的客户查询等待时间的中值。

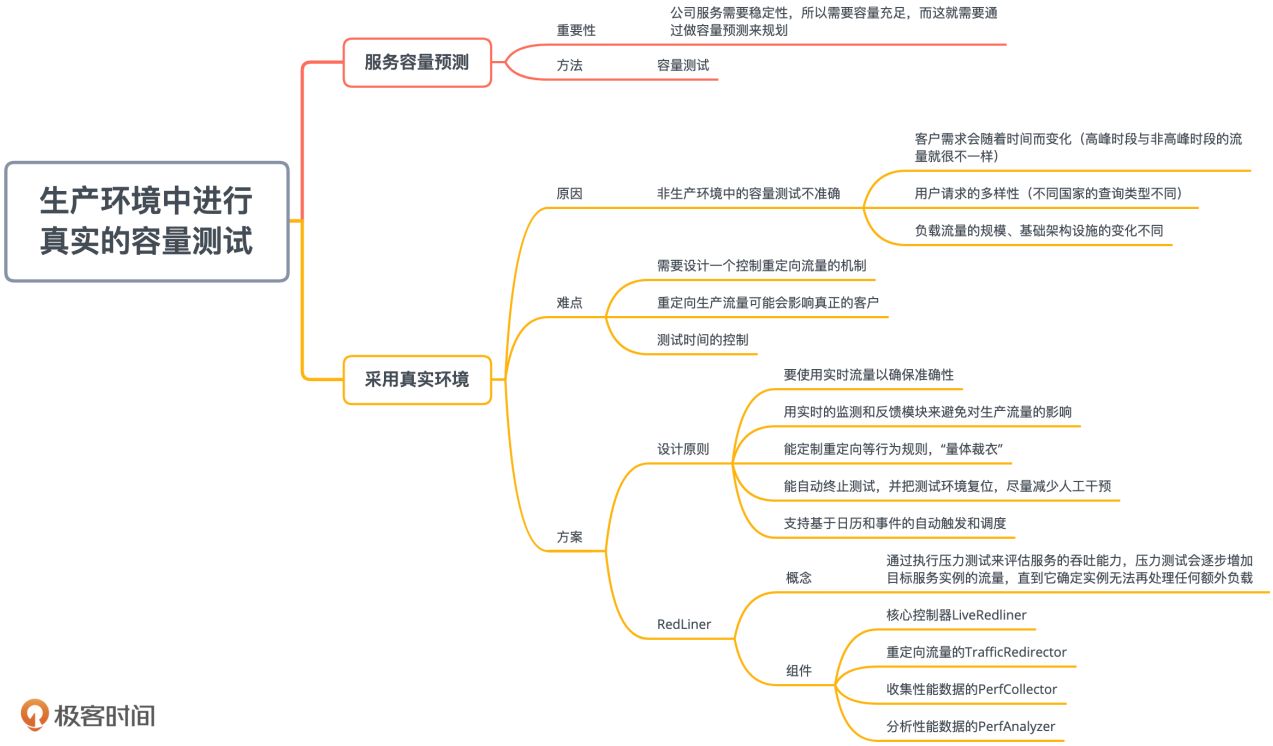
比较这两幅图，你可以看到，开始测试的阶段，RedLiner 不断提高重定向百分比，实际的流量持续增加；同时客户感受的查询等待时间也慢慢加大。

等到 SUT 不堪重负时，查询等待时间也就太大了，超出了客户能接受的阈值（40 毫秒）。所以，RedLiner 决定逐步降低重定向百分比，最后重置到初始状态。

总结

我们这一讲介绍了一个在生产环境中，进行真实场景压力和容量测试的方案，这里面的关键点，是**逐步而智能地把一部分流量重定向到被测试的系统上面。**

这个案例是我们在领英的生产实践，但我觉得在你的公司里实现这么一个类似的系统一点也不难。希望这些分享能帮助你设计和实现。



唐代诗人白居易的《长相思》最后的几句是：“愿作远方兽，步步比肩行。愿作深山木，枝枝连理生。”说的是主人公愿意与心爱的人相守到老，哪怕是做山林中的野兽和树木，一起步步连心，亦步亦趋，比肩而行，并肩而居。

我们用实际的生产负载做容量测试时，要小心控制重定向的流量“步距”大小，尽量“小步勤挪”，并且实时地观测，才不会影响客户的体验，并且得到比较准确的结果。

思考题

你们公司有没有类似的解决方案，来准确地测量一个服务需要的容量呢？如果有，和我讲的具体方案有何异同？不同的地方是基于什么考虑呢？

如果没有，你可以考虑实现一个，我相信一定会让老板对你另眼相看的。

欢迎你在留言区分享自己的思考，与我和其他同学一起讨论，也欢迎你把文章分享给自己的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　24 | 跨层案例：如何优化程序、OS和存储系统的交互？

下一篇　26 | 怎么规划和控制数据库的复制延迟大小？

精选留言 (2)

写留言

许童童
2020-01-22

有个问题，生产环境流量不多或者根本就没有流量，这个时候怎么办呢？

作者回复: 如果说实际生产流量不够驱动一个SUT，这种情况就只能用模拟流量了（比如人工产生流量去驱动）。

2

黄海峰

2020-01-22

“模仿实现一个一点也不难”。。。我被高估了，后面加餐希望多点技术细节介绍

作者回复: :)。这个具体实现要根据实际的系统，因系统而异。关键是搞清楚负载均衡（就是来了一个流量，让哪个服务器处理），然后修改均衡机制（就是让它“不均衡”）。要小心别弄砸，因为负载均衡很关键，而且最好自动化。